AGGREGATION AND COMPOSITION

How to Identify Composition : Guideline

Consider showing composition when:

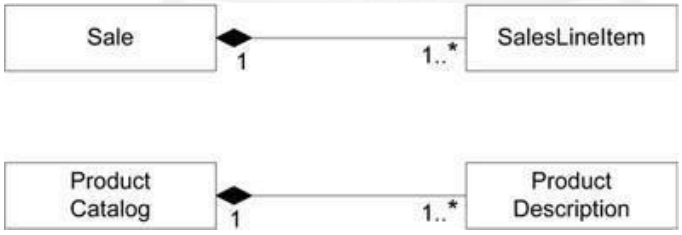
- The lifetime of the part is bound within the lifetime of the composite there is a create-delete dependency of the part on the whole.
- There is an obvious whole-part physical or logical assembly.
- Some properties of the composite propagate to the parts, such as the location.
- Operations applied to the composite propagate to the parts, such as destruction, movement, and recording.

Composition in the NextGen Domain Model

In the POS domain, the SalesLineItems may be considered a part of a composite

Sale;

Aggregation in the point-of-sale application.



SYSTEM SEQUENCE DIAGRAMS

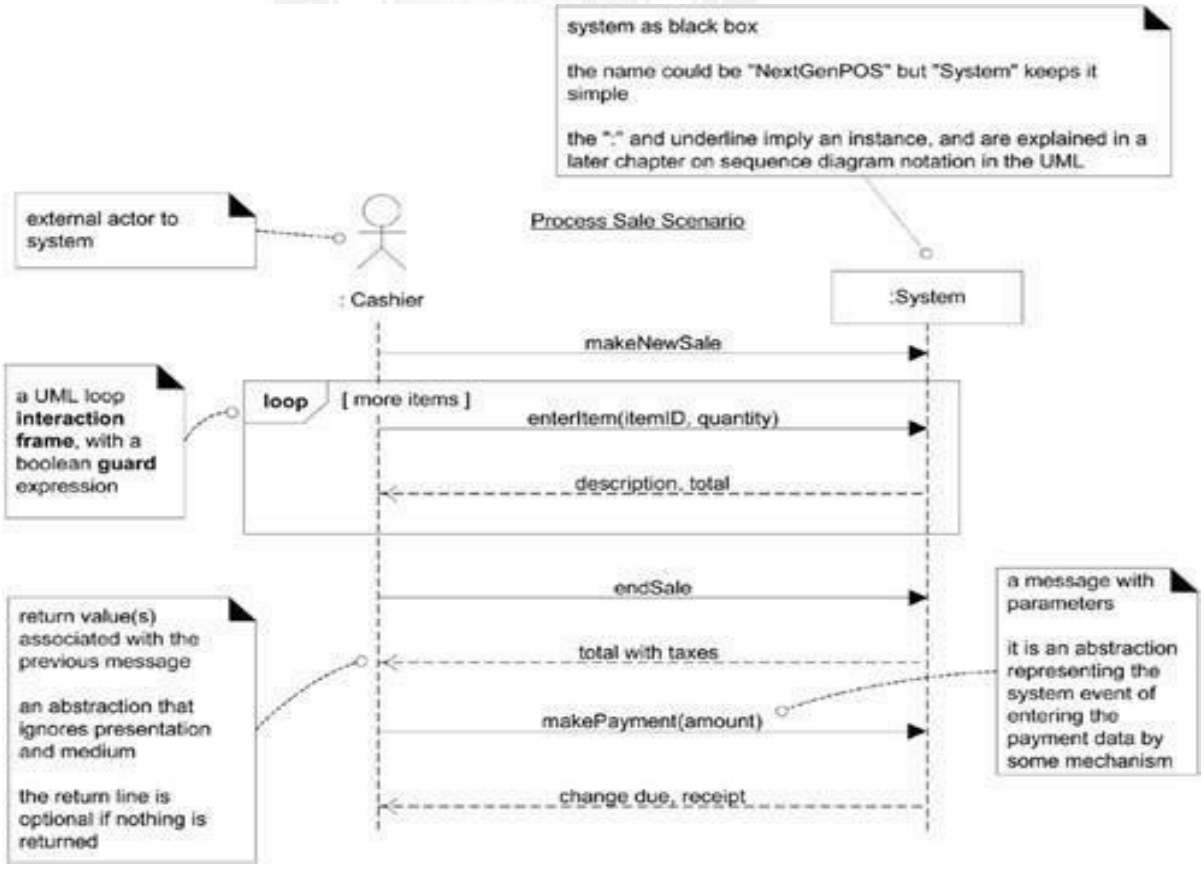
Use cases describe how external actors interact with the software system we are interested in creating. During this interaction an actor generates system events to a system, usually requesting some system operation to handle the event.

For example, when a cashier enters an item's ID, the cashier is requesting the POS system to record that item's sale (the enterItem event). That event initiates an operation upon the system. The use case text implies the enterItem event, and the SSD makes it concrete and explicit.

A system sequence diagram is a picture that shows, for one particular scenario of a use case, the events that external actors generate their order, and inter- system events. All systems are treated as a black box.

Guideline : Draw an SSD for a main success scenario of each use case, and frequent or complex alternative scenarios.

SSD for a Process Sale scenario.



Why to Draw an SSD?

A software system reacts to three things:

1) external events from actors (humans or computers),

2) timer events,

3) faults or exceptions (which are often from external sources).

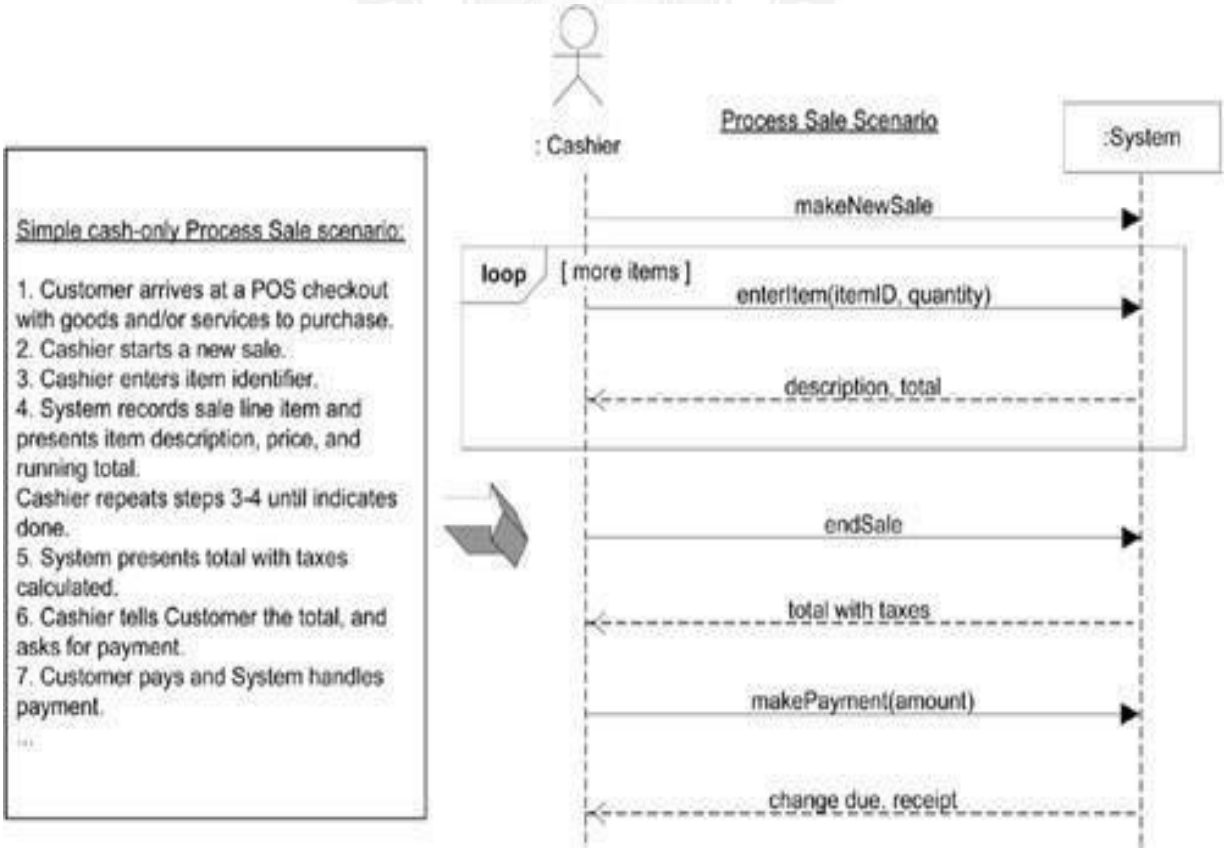
Therefore, it is useful to know what, precisely, are the external input events the system events. They are an important part of analyzing system behavior.

System behavior is a description of what a system does, without explaining how it does it. One part of that description is a system sequence diagram.

RELATIONSHIP BETWEEN SSDS AND USE CASES

An SSD shows system events for one scenario of a use case, therefore it is generated from inspection of a use case (see Figure below).

SSDs are derived from use cases; they show one scenario.



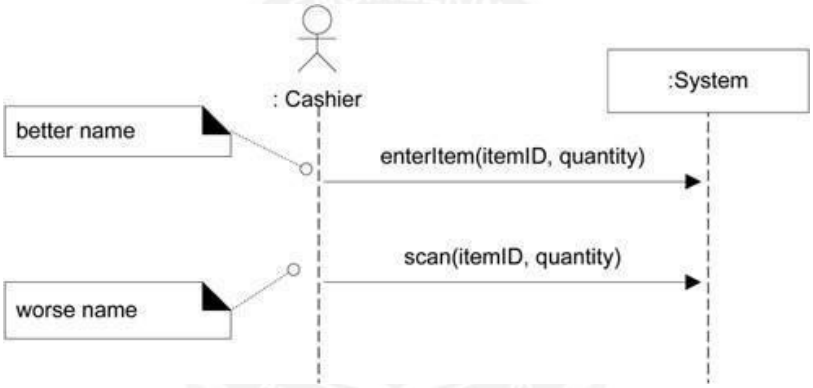
How to Name System Events and Operations?

Which is better, scan(itemID) or enterItem(itemID)?

System events should be expressed at the abstract level of intention rather than in terms of the physical input device.

Thus "enterItem" is better than "scan" (that is, laser scan) because it captures the intent of the operation while remaining abstract and noncommittal with respect to design choices about what interface is used to capture the system event.

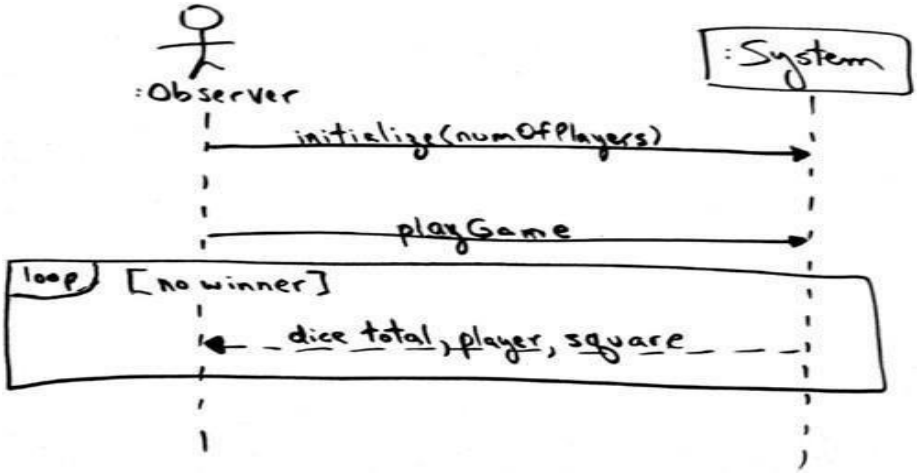
Choose event and operation names at an abstract level.



Example: Monopoly SSD

The Play Monopoly Game use case is simple, as is the main scenario. The observing person initializes with the number of players, and then requests the simulation of play, watching a trace of the output until there is a winner.

SSD for a Play Monopoly Game scenario.



Process:

Draw SSDs only for the scenarios chosen for the next iteration. Don't create SSDs for all scenarios, unless you are using an estimation technique that requires identification of all system operations.